```
ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE GD
OBJECT MODULE PLACED IN GENDEF.OBJ
COMPILER INVOKED BY:  :F0:GENDEF.PLM DEBUG XREF OPTIMIZE(2) DATE(APRIL 9, 1982)


  1          gd:
             do;
             /* l i b r a r y   g e n e r a t o r   f o r   D I S K D E F */

             $include (:f1:lit.plb)
         =   /* literal declarations for basic tokens */
  2   1  =   declare
         =       lit literally 'literally',
         =       tiden  lit '01', tnumb  lit '02', tstrng lit '03', tspecl lit '04',
         =
         =       /* token# is the base token number for each set of tokens */
         =       token2 lit '140', token3 lit '160', token4 lit '180',
         =       token5 lit '200', token6 lit '210', token7 lit '220',
         =       token8 lit '230',
         =       tokenm lit '7',
         =
         =       /* single character tokens ! ' # $ % & ( ) * + , - * / : ; < = > ?
         =       are all represented by their ascii values */
         =
         =       /* special two character sequences (started by single character) */
         =
         =       /* two character tokens: */
         =
         =       /* three character tokens: */
         =
         =       /* four character tokens: */
         =
         =       /* five character tokens: */
         =       tdisks lit '200', tendef lit '201',
         =
         =       /* six character tokens: */
         =       tmaclib lit '210',
         =
         =       /* seven character tokens: */
         =       tdiskdef lit '220';

  3   1      declare
                 true    lit '1',
                 false   lit '0',
                 forever lit 'while true',
                 cr      lit '0dh',
                 tab     lit '09h';

             /* global procedures */
  4   1      boot: procedure external;
  5   2          end boot;

  6   1      printchar: procedure(c) external;
  7   2          declare c byte;
  8   2          end printchar;
```

CP/M-86 v.1.1 (Vol. II)
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # C81-162

```
 9  1     print: procedure(a) external;
10  2         declare a address;
11  2         end print;

12  1     crlf: procedure external;
13  2         end crlf;

14  1     abort: procedure(msg) external;
15  2         declare msg address;
16  2         end abort;

17  1     scan: procedure external;
18  2         end scan;

19  1     scan$ini: procedure external;
20  2         end scan$ini;

21  1     setup: procedure external;
              /* set up the input file on each pass */
22  2         end setup;

23  1     putline: procedure external;
24  2         end putline;

25  1     writerrs: procedure external;
26  2         end writerrs;

27  1     errptr: procedure(msg) external;
28  2         declare msg address;
29  2         end errptr;

              /* code emitters */
30  1     emitbyte: procedure(b) external;
31  2         declare b byte;
32  2         end emitbyte;

33  1     emitcrlf: procedure external;
34  2         end emitcrlf;

35  1     emitoptab: procedure(a) external;
36  2         declare a address;
37  2         end emitoptab;

38  1     emitcomma: procedure external;
39  2         end emitcomma;

40  1     emitaddr: procedure(a) external;
41  2         declare a address;
42  2         end emitaddr;

43  1     emitchar: procedure(a) external;
44  2         declare a address;
45  2         end emitchar;

46  1     emitline: procedure external;
47  2         end emitline;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
48  1     emitcharn: procedure(a,n) external;
49  2         declare a address, n byte;
                  /* symbol, followed by two digit n */
50  2         end emitcharn;

51  1     emitcomment: procedure(a) external;
52  2         declare a address;
53  2         end emitcomment;

54  1     emitdec8: procedure(b) external;
55  2         declare b byte;
56  2         end emitdec8;

57  1     emitdec16: procedure(w) external;
58  2         declare w address;
59  2         end emitdec16;

60  1     emitdecb: procedure(w) external;
61  2         declare w address;
62  2         end emitdecb;

63  1     emitdw: procedure external;
64  2         end emitdw;

65  1     emitdb: procedure external;
66  2         end emitdb;

67  1     emitdwn: procedure(n) external;
68  2         declare n address;
69  2         end emitdwn;

70  1     emitdbn: procedure(n) external;
71  2         declare n byte;
72  2         end emitdbn;

73  1     emitdwnc: procedure(n,c) external;
74  2         declare (n,c) address;
75  2         end emitdwnc;

76  1     emitdbnc: procedure(n,c) external;
77  2         declare n byte, c address;
78  2         end emitdbnc;

79  1     efinis: procedure external;
80  2         end efinis;

81  1     emittab: procedure external;
82  2         end emittab;

83  1     emitnulcom: procedure external;
84  2         end emitnulcom;

           /* global variables */
85  1     declare
           parm (26) byte external,
           comm$tog  byte at (.parm(2)),  /* disk comment */
           off$tog   byte at (.parm(14)), /* generate 'offset' */
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
            scan$tog  byte at (.parm(18)), /* scanner trace */
            z80$tog   byte at (.parm(25)); /* z80, 8080, 8085 mode */

 86  1      declare
            errset  byte external,
            eofset  byte external,
            cbuff(8) byte external,
            value   address external,
            nextc   byte external,
            token   byte external,
            continue byte external,
            acclen  byte external,
            accum(32) byte external;

          /* code emitters */
 87  1      setemit: procedure external;
 88  2          end setemit;

 89  1      prnib: procedure(d);
 90  2          declare d byte;
             /* print nibble d */
 91  2          if d > 9 then
 92  2              d = 'A' + d - 10; else
 93  2              d = d + '0';
 94  2          call printchar(d);
 95  2          end prnib;

 96  1      prhex: procedure(h);
 97  2          declare h byte;
 98  2          call prnib(shr(h,4));
 99  2          call prnib(h and 0fh);
100  2          end prhex;

101  1      praddr: procedure(a);
102  2          declare a address;
103  2          call prhex(high(a));
104  2          call prhex(low(a));
105  2          end praddr;

106  1      prdig: procedure(d);
107  2          declare d byte;
             /* print the decimal number given by d */
108  2          call printchar('0'+d);
109  2          end prdig;

110  1      prdec: procedure(b);
111  2          declare b byte;
             /* print the decimal value of b */
112  2          call prdig(b/100);
113  2          call prdig(b/10 mod 10);
114  2          call prdig(b mod 10);
115  2          end prdec;

          /* trace functions */

116  1      declare
             ident    lit '1',
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                number  lit '2',
                string  lit '3',
                special lit '4';

117   1     trace: procedure(msga);
118   2         declare
                    msga address,
                    (i,c) byte;
119   2         if scan$tog then
120   2             do;
121   3             call putline;
122   3             call print(msga);
123   3             call printchar(',');
124   3                 do i = 1 to acclen;
125   4                 if (c:=accum(i-1)) < ' ' then
126   4                     do;
127   5                     call printchar('$');
128   5                     call prhex(c);
129   5                     end; else
130   4                 call printchar(c);
131   4                 end;
132   3             call printchar(':');
133   3             call prdec(token);
134   3             if token = number then
135   3                 do;
136   4                 call printchar(',');
137   4                 call praddr(value);
138   4                 call printchar('H');
139   4                 end;
140   3             if eofset then
141   3                 call print(.('End-File$'));
142   3             end;
143   2         end trace;

              /* parsing procedures for non terminals */

144   1     bypass: procedure(x) byte;
145   2         declare x byte;
146   2         if token = x then
147   2             do;
148   3             call scan;
149   3             return true;
150   3             end;
151   2         return false;
152   2         end bypass;

153   1     declare eltinx byte;   /* index into set when 'element' returns true */

154   1     element: procedure(list) byte;
155   2         declare list address; /* address of set to search */
                  /* element searches for the current token in the set of
                  tokens addressed by 'list' and returns true if item is
                  found.  as a side-effect, 'eltinx' is set to the item's index */
156   2         declare set based list (1) byte, x byte;
157   2         eltinx = -1;
158   2             do forever;
159   3             if (x := set(eltinx:=eltinx+1)) = 0 then return false;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
161  3            if token = x then return true;
163  3            end;
164  2        end element;

165  1    endline: procedure byte;
              /* return true if end of line or file */
166  2        return (token = cr) or eofset;
167  2        end endline;

168  1    recover: procedure;
169  2            do while not endline;
170  3            call scan;
171  3            end;
172  2        end recover;

173  1    emitdollar: procedure;
174  2        if not z80$tog then
175  2            call emitchar(.('offset $'));
176  2        call emitbyte('$');
177  2        end emitdollar;

178  1    declare
              /* diskdef parameter values */
              maxdisks lit '16',
              ndisks   byte,          /* number of disks */
              diskset  (maxdisks) byte; /* true if included in diskdef */

179  1    disks: procedure;
              /* handle the disks n macro */
180  2        declare i byte;
181  2        if ndisks > 0 then
182  2           ndisks = 0; else
183  2        if token = number then
184  2            ndisks = value; else
185  2            ndisks = 0;
186  2        if ndisks = 0 or ndisks > maxdisks then
187  2            call errptr(.('bad val$'));
188  2        call scan;
              /* dpbase equ $ */
189  2        call emitchar(.('dpbase$'));
190  2        call emitoptab(.('equ$'));
191  2        if off$tog then
192  2            call emitdollar; else
193  2            call emitbyte('$');
194  2        call emitcomment(.('Base of Disk Parameter Blocks$'));
195  2            do i = 1 to ndisks;
              /* dpe00: dw xlt00,0000h */
196  3            call emitcharn(.('dpe$'),i-1);
197  3            if z80$tog then
198  3                call emitbyte(':');
199  3            call emitdw;
200  3            call emitcharn(.('xlt$'),i-1);
201  3            call emitcomma;
202  3            call emitaddr(0);
203  3            call emitcomment(.('Translate Table$'));
              /* dw 0000h,0000h */
204  3            call emitdw;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
205  3            call emitaddr(0);
206  3            call emitcomma;
207  3            call emitaddr(0);
208  3            call emitcomment(.('Scratch Area$'));
                  /* dw dirbuf,dpb00 */
209  3            call emitdw;
210  3            call emitchar(.('dirbuf,$'));
211  3            call emitcharn(.('dpb$'),i-1);
212  3            call emitcomment(.('Dir Buff, Parm Block$'));
                  /* dw csv00,alv00 */
213  3            call emitdw;
214  3            call emitcharn(.('csv$'),i-1);
215  3            call emitcomma;
216  3            call emitcharn(.('alv$'),i-1);
217  3            call emitcomment(.('Check, Alloc Vectors$'));
218  3            end;
219  2        end disks;

220  1    declare
           npar lit '10',
           dn     lit '0',  /* disk number */
           fsc    lit '1',  /* first sector */
           lsc    lit '2',  /* last sector */
           skf    lit '3',  /* skew factor */
           bls    lit '4',  /* block size */
           dks    lit '5',  /* disk size */
           dir    lit '6',  /* directory size */
           cks    lit '7',  /* check sum size */
           ofs    lit '8',  /* offset tracks */
           com    lit '9';  /* CP/M 1.4 compatibility flag */

221  1    declare
           lower (*) address data
          /* dn    fsc    lsc    skf    bls  */
           (0000, 0000, 0001, 0000, 1024,
          /* dks   dir    cks    ofs    com  */
            0001, 0001, 0000, 0000, 0000),
           upper (*) address data
          /* dn    fsc    lsc    skf    bls  */
           (0015, 0015, 65535, 0255, 16384,
          /* dks   dir    cks    ofs    com  */
            65535, 65534, 65534, 65535, 0000),
           optional (*) byte data
          /* dn    fsc    lsc    skf    bls  */
           (false, false, false, true,  false,
          /* dks   dir    cks    ofs    com  */
            false, false, false, false, true);

222  1    declare
           dsmax (*) address data
           /*        value of bls          */
           /* 1024  2048  4096  8192  16384 */
           (  8192, 4096, 2048, 1024, 512  );

223  1    declare
           bsh (*) byte data
           /* 1024  2048  4096  8192  16384 */
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                    (  3,   4,   5,   6,   7  ),
                    blm (*) byte data
                    /* 1024  2048  4096  8192  16384 */
                    (  7,   15,  31,  63,  127 ),
                    dsm (*) byte data
                    /* 1024  2048  4096  8192  16384 */
                    (  0,   1,   3,   7,   15 );

224   1         declare
                    ddir (*) address data
                    /* 1024  2048  4096  8192  16384 */
                    (  32,  64,  128, 256, 512  );

225   1         range: procedure;
226   2             call errptr(.('range  $'));
227   2             end range;

228   1         diskdef: procedure;
                    /* handle diskdef, first item scanned */
229   2             declare
                        pv  (npar) address,
                        parmissing (npar) byte,
                        (pc, i)  byte,
                        (d, v) address;
230   2             declare
                        sectors address,
                        (nxtsec, nxtbas, neltst, nelts) address;

231   2             gcd: procedure(m,n) address;
232   3                 declare (m,n,r) address;
233   3                     do forever;
234   4                     r = m mod n;
235   4                     if r = 0 then
236   4                         return n;
237   4                     m = n;
238   4                     n = r;
239   4                     end;
240   3                 end gcd;

241   2             equiv: procedure(a,b);
242   3                 declare (a,b) address;
                        /* dpb1 equ dpb0 */
243   3                 call emitcharn(a,pv(0));
244   3                 call emitoptab(.('equ$'));
245   3                 call emitcharn(a,pv(1));
246   3                 call emitcomment(b);
247   3                 end equiv;

248   2                 comment: procedure(v,a);
249   3                     declare (v, a) address;
                            /* write diskdef comment line */
250   3                     call emitbyte(';');
251   3                     call emittab;
252   3                     call emitdecb(v);
253   3                     call emitbyte(':');
254   3                     call emittab;
255   3                     call emitchar(a);
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
256  3              call emitcrlf;
257  3              end comment;

258  2          do i = 0 to last(parmissing);
259  3          parmissing(i) = true;
260  3          end;
261  2        pc = 0;
262  2        token = ',';
263  2          do while not (endline or pc = npar);
264  3          call trace(.('   diskparm$'));
265  3          pv(pc) = 0;
266  3          if not bypass(',') then
267  3              call errptr(.('delimit$')); else
268  3          if (parmissing (pc) !=
                   (token = cr or token = ',')) then
269  3              do;
270  4              if not optional(pc) then
271  4                  call errptr(.('missing$'));
272  4              end; else
273  3              do;
274  4              if token <> number then
275  4                  call errptr(.('numeric$')); else
276  4                  do;
                       /* numeric value, check range */
277  5                  if value < lower(pc) or
                          value > upper(pc) then
278  5                      call range; else
                       /* make special range checks */
279  5                  if pc = dn then
280  5                      do;
281  6                      if diskset(value) then
282  6                          call errptr(.('duplic $'));
283  6                      diskset(value) = true;
284  6                      end; else
285  5                  if pc = lsc and value <= pv(fsc) then
286  5                      call range; else
287  5                  if pc = skf and
                           value > (pv(lsc) - pv(fsc)) then
288  5                      call range; else
289  5                  if pc = bls then
290  5                      do;
291  6                      if value = 1024  then value = 0; else
293  6                      if value = 2048  then value = 1; else
295  6                      if value = 4096  then value = 2; else
297  6                      if value = 8192  then value = 3; else
299  6                      if value = 16384 then value = 4; else
301  6                          call range;
302  6                      end; else
303  5                  if pc = dks then
304  5                      do;
305  6                      if value > 256 and pv(bls) = 0 then
306  6                          call range; else
307  6                      if value > dsmax(pv(bls)) then
308  6                         call range;
                           end; else
310  5                   if pc = dir then
311  5                       do;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
312  6                    if (value and 11b) <> 0 then
313  6                        call range; else
314  6                        do;
                              /* compute alloc vector */
315  7                        v = 0;
316  7                        d = value;
                                  /* loop until zero or negative */
317  7                            do while not rol(high(d-1),1);
318  8                            d = d - ddir(pv(bls));
319  8                            if low(v) then
320  8                                do;
321  9                                call errptr(.('Alloc  $'));
322  9                                d = 0;
323  9                                end; else
324  8                            v = shr(v,1) or 8000h;
325  8                            end;
326  7                        end;
327  6                    end; else
328  5                if pc = cks and ((value and 11b) <> 0 or
                                  value > pv(dir)) then
329  5                    call range;
                         pv(pc) = value;
331  5                    end;
332  4                call scan;
333  4                end;
334  3            if errset then
335  3                return;
336  3            pc = pc + 1;
337  3            end;
                 /* check for abbreviated form */
338  2           if not endline then
339  2               return;
340  2           if pc = 2 then
341  2               do;
342  3               if not diskset(pv(1)) then
343  3                   call errptr(.('no disk$'));
344  3               if comm$tog then
345  3                   do;
346  4                   call emitnulcom;
347  4                   call emitbyte(';');
348  4                   call emittab;
349  4                   call emitchar(.('Disk $'));
350  4                   call emitdec8(pv(0));
351  4                   call emitchar(.(' is the same as Disk $'));
352  4                   call emitdec8(pv(1));
353  4                   call emitcrlf;
354  4                   call emitnulcom;
355  4                   end;
356  3               call equiv(.('dpb$'),.('Equivalent Parameters$'));
357  3               call equiv(.('als$'),.('Same Allocation Vector Size$'));
358  3               call equiv(.('css$'),.('Same Checksum Vector Size$'));
359  3               call equiv(.('xlt$'),.('Same Translate Table$'));
360  3               return;
361  3               end; else
362  2            if pc < npar-1 then
363  2               do;
364  3               call errptr(.('too few$'));
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
365  3          return;
366  3          end;
                /* write general disk statistics */
                sectors = pv(lsc) - pv(fsc) + 1;
368  2          if comm$tog then
369  2              do;
                    /* ;<crlf>;<tab>Disk 000 is CP/M 1.4 Single(/Double) .. */
370  3              call emitnulcom;
371  3              if not parmissing(com) then
372  3                  do;
373  4                  call emitbyte(';');
374  4                  call emittab;
375  4                  call emitchar(.('Disk $'));
376  4                  call emitdec8(pv(dn));
377  4                  call emitchar(
                            .(' is CP/M 1.4 Double Density Compatible$'));
378  4                  call emitcrlf;
379  4                  end;
                    /* pv(dks) * 8/16/32/64/128 for 1k/2k/4k/8k/16k */
380  3              d = pv (dks) * shl(100b,pv(bls)+1);
381  3              call emitbyte(';');
382  3              call emittab;
383  3              if d > 0 then
384  3                  call emitdecb(d); else
385  3                  call emitchar(.('65536$'));
386  3              call emitbyte(';');
387  3              call emittab;
388  3              call emitchar(.('128 Byte Record Capacity$'));
389  3              call emitcrlf;
                    /* pv(dks) * 1k/2k/4k/8k/16k */
390  3              d = 1;
391  3              if pv(bls) > 0 then
392  3                  d = shl(d,pv(bls));
393  3              call comment(pv(dks) * d,.('Kilobyte Drive  Capacity$'));
394  3              call comment(pv(dir),.('32 Byte Directory Entries$'));
395  3              call comment(pv(cks),.('Checked Directory Entries$'));
396  3              if not parmissing(com) then
397  3                  d = 128; else
398  3                  do;
                        /* extents are folded */
399  4                  if pv(dks) > 256 then
400  4                      d = 32; else d = 64;
402  4                  d = shl(d,pv(bls)+1);
403  4                  end;
404  3              call comment(d,.('Records / Extent$'));
405  3              call comment(shl(100b,pv(bls)+1),
                        .('Records / Block$'));
406  3              call comment(sectors,.('Sectors / Track$'));
407  3              call comment(pv(ofs),.('Reserved  Tracks$'));
408  3              if pv(skf) = 0 then parmissing(skf) = true;
410  3              if not parmissing(skf) then
411  3                  call comment(pv(skf),.('Sector Skew Factor$'));
412  3              call emitnulcom;
413  3              end;
                /* dpb0 equ $ */
414  2          call emitcharn(.('dpb$'),pv(dn));
415  2          call emitoptab(.('equ$'));
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
416  2          call emitdollar;
417  2          call emitcomment(.('Disk Parameter Block$'));
418  2          call emitdwnc(sectors,
                    .('Sectors Per Track$'));
419  2          call emitdbnc(bsh(pv(bls)),
                    .('Block Shift$'));
420  2          call emitdbnc(blm(pv(bls)),
                    .('Block Mask$'));
421  2          i = pv(bls);
422  2          if pv(dks) > 256 then
423  2              i = i - 1;
424  2          if parmissing(com) then
425  2              call emitdbnc(dsm(i),.('Extnt Mask$')); else
426  2              call emitdbnc(pv(com),.('1.4 Compatible$'));
427  2          call emitdwnc(pv(dks)-1,.('Disk Size - 1$'));
428  2          call emitdwnc(pv(dir)-1,.('Directory Max$'));
                /* compute allocation vector initialization */
429  2          call emitdbnc(high(v),.('Alloc0$'));
430  2          call emitdbnc(low(v),.('Alloc1$'));
431  2          call emitdwnc(pv(cks)/4,.('Check Size$'));
432  2          call emitdwnc(pv(ofs),.('Offset$'));

                /* generate allocation vector */
433  2          call emitcharn(.('xlt$'),pv(dn));
434  2          call emitoptab(.('equ$'));
435  2          if parmissing(skf) then
436  2              do;
437  3              call emitdec8(0);
438  3              call emitcomment(.('No Translate Table$'));
439  3              end; else
440  2              do;
441  3              call emitdollar;
442  3              call emitcomment(.('Translate Table$'));
443  3              nxtsec, nxtbas = 0;
444  3              neltst = sectors/gcd(sectors,pv(skf));
445  3              nelts = neltst;
446  3                  do d = 0 to sectors-1;
447  4                  if d mod 4 <> 0 then
448  4                      call emitcomma; else
449  4                      do;
450  5                      if d > 0 then
451  5                          call emitcrlf;
452  5                      if sectors < 256 then
453  5                          call emitdb; else
454  5                          call emitdw;
455  5                      end;
456  4                  call emitdec16(nxtsec+pv(fsc));
457  4                  nxtsec = nxtsec + pv(skf);
458  4                  if nxtsec >= sectors then
459  4                      nxtsec = nxtsec - sectors;
460  4                  if (nelts := nelts - 1) = 0 then
461  4                      do;
462  5                      nxtsec = (nxtbas := nxtbas + 1);
463  5                      nelts = neltst;
464  5                      end;
465  4                  end;
466  3              call emitcrlf;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
467  3              end;

                 /* generate allocation vector size */
468  2           call emitcharn(.('als$'),pv(dn));
469  2           call emitoptab(.('equ$'));
470  2           d = pv(dks)/8;
471  2           if pv(dks) mod 8 <> 0 then
472  2              d = d + 1;
473  2           call emitdec16(d);
474  2           call emitcomment(.('Allocation Vector Size$'));

                 /* generate checksum vector size */
475  2           call emitcharn(.('css$'),pv(dn));
476  2           call emitoptab(.('equ$'));
477  2           call emitdec16(pv(cks)/4);
478  2           call emitcomment(.('Check Vector Size$'));
479  2           end diskdef;

480  1       endef: procedure;
                 /* generate end of disk def code */
481  2           declare i byte;
482  2           reserve: procedure;
483  3              if z80$tog then
                       /* $z set for z80, 8080, or 8085 */
484  3                 call emitoptab(.('ds$')); else
485  3                 call emitoptab(.('rs$'));
486  3              end reserve;
487  2           call emitnulcom;
488  2           call emitbyte(';');
489  2           call emittab;
490  2           call emitchar(.('Uninitialized Scratch Memory Follows:$'));
491  2           call emitcrlf;
492  2           call emitnulcom;

                 /* begdat equ $ */
493  2           call emitchar(.('begdat$'));
494  2           call emitoptab(.('equ$'));
495  2           call emitdollar;
496  2           call emitcomment(.('Start of Scratch Area$'));

                 /* dirbuf: ds/rs 128 */
497  2           call emitchar(.('dirbuf$'));
498  2           if z80$tog then
499  2              call emitbyte(';');
500  2           call reserve;
501  2           call emitdec8(128);
502  2           call emitcomment(.('Directory Buffer$'));

                 /* alv0: ds als0, csv0: ds css0 */
503  2              do i = 0 to last(diskset);
504  3              if diskset(i) then
505  3                 do;
506  4                 call emitcharn(.('alv$'),i);
507  4                 if z80$tog then
508  4                    call emitbyte(';');
509  4                 call reserve;
510  4                 call emitcharn(.('als$'),i);
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
511  4                call emitcomment(.('Alloc Vector$'));
512  4                call emitcharn(.('csv$'),i);
513  4                if z80$tog then
514  4                    call emitbyte('!');
515  4                call reserve;
516  4                call emitcharn(.('css$'),i);
517  4                call emitcomment(.('Check Vector$'));
518  4                end;
519  3            end;

                 /* enddat equ $, datsiz equ $-begdat */
520  2            call emitchar(.('enddat$'));
521  2            call emitoptab(.('equ$'));
522  2            call emitdollar;
523  2            call emitcomment(.('End of Scratch Area$'));
524  2            call emitchar(.('datsiz$'));
525  2            call emitoptab(.('equ$'));
526  2            call emitdollar;
527  2            call emitchar(.('-begdat$'));
528  2            call emitcomment(.('Size of Scratch Area$'));
529  2            call emitdbnc(0,.('Marks End of Module$'));
530  2            end endef;

531  1        program: procedure;
                     /* handle the entire genlib program */
532  2            declare
                         i byte;
533  2            do i = 0 to last(diskset);
534  3            diskset(i) = false;
535  3            end;
536  2            ndisks = 0;
537  2            do while not eofset;
538  3            errset = false;
539  3            call scan;
540  3            call emitline;
541  3            call trace(.('program$'));
542  3            if token <> cr then
543  3            if not element(.(
                         tmaclib,
                         tdisks,
                         tdiskdef,
                         tendef,
                         0)) then
544  3                call errptr(.('No Stmt$')); else
545  3                do;
546  4                do case eltinx;
                         /* maclib */
547  5                    do;
548  6                    call scan;
549  6                    call trace(.('  maclib$'));
550  6                    call scan;
551  6                    call trace(.('  macpar$'));
552  6                    end;
                         /* disks */
553  5                    do;
554  6                    call scan;
555  6                    call trace(.(' disks$'));
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
356  6                  call disks;
357  6                  end;
                      /* diskdef (10 parameters) */
358  5                  do;
359  6                  call trace(.(' diskdef$'));
360  6                  call diskdef;
561  6                  end;
                      /* endef */
562  5                  do;
563  6                  call scan;
564  6                  call endef;
565  6                  end;
566  5                end;
567  4              if token <> cr then
568  4                 call errptr(.('extra  $'));
569  4              end;
570  3            call recover;
571  3            end;
572  2        end program;

573  1    plm: procedure public;
              /* main program, called from transient interface */
574  2        call setemit; /* setup the code emitters */
575  2        call scan$ini; /* initialize the scanner */
576  2        call setup; /* set up the input file */
577  2        call program;
578  2        call writerrs;
579  2        call efinis;     /* close the code file */
580  2        call boot;
581  2        end;
582  1    end;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
40	0000H	2	A. . . . . WORD PARAMETER 41
51	0000H	2	A. . . . . WORD PARAMETER 52
48	0000H	2	A. . . . . WORD PARAMETER 49
248	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 249 255
241	0006H	2	A. . . . . WORD PARAMETER AUTOMATIC 242 243 245
9	0000H	2	A. . . . . WORD PARAMETER 10
101	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 102 103 104
43	0000H	2	A. . . . . WORD PARAMETER 44
35	0000H	2	A. . . . . WORD PARAMETER 36
14	0000H		ABORT. . . . . PROCEDURE EXTERNAL(4) STACK=0000H
86	0000H	1	ACCLEN . . . . . BYTE EXTERNAL(40) 124
86	0000H	32	ACCUM. . . . . BYTE ARRAY(32) EXTERNAL(41) 125
30	0000H	1	B. . . . . BYTE PARAMETER 31
241	0004H	2	B. . . . . WORD PARAMETER AUTOMATIC 242 246
110	0004H	1	B. . . . . BYTE PARAMETER AUTOMATIC 111 112 113 114
54	0000H	1	B. . . . . BYTE PARAMETER 55
223	004BH	5	BLM. . . . . BYTE ARRAY(5) DATA 420
220			BLS. . . . . LITERALLY 289 305 307 318 380 391 392 402 405 419 420 421
4	0000H		BOOT . . . . . PROCEDURE EXTERNAL(0) STACK=0000H 580
223	0046H	5	BSH. . . . . BYTE ARRAY(5) DATA 419
144	012DH	24	BYPASS . . . . . PROCEDURE BYTE STACK=0006H 266
118	0025H	1	C. . . . . BYTE 125 128 130
6	0000H	1	C. . . . . BYTE PARAMETER 7
76	0000H	2	C. . . . . WORD PARAMETER 77
73	0000H	2	C. . . . . WORD PARAMETER 74
86	0000H	8	CBUFF. . . . . BYTE ARRAY(8) EXTERNAL(35)
220			CKS. . . . . LITERALLY 328 395 431 477
220			COM. . . . . LITERALLY 371 396 424 426
248	0957H	40	COMMENT. . . . . PROCEDURE STACK=000AH 393 394 395 404 405 406 407 411
85	0002H	1	COMMTOG. . . . . BYTE EXTERNAL(32) AT 344 368
86	0000H	1	CONTINUE . . . . . BYTE EXTERNAL(39)
3			CR . . . . . LITERALLY 166 268 542 567
12	0000H		CRLF . . . . . PROCEDURE EXTERNAL(3) STACK=0000H
106	0004H	1	D. . . . . BYTE PARAMETER AUTOMATIC 107 108
229	0014H	2	D. . . . . WORD 316 317 318 322 380 383 384 390 392 393 397 400 401 402 404 446 447 450 470 472 473
89	0004H	1	D. . . . . BYTE PARAMETER AUTOMATIC 90 91 92 93 94
224	0032H	10	DDIR . . . . . WORD ARRAY(5) DATA 318
220			DIR. . . . . LITERALLY 310 328 394 428
228	02D8H	1581	DISKDEF. . . . . PROCEDURE STACK=001EH 560
179	01BBH	273	DISKS. . . . . PROCEDURE STACK=000AH 556
178	0029H	16	DISKSET. . . . . BYTE ARRAY(16) 281 283 342 503 504 533 534
220			DKS. . . . . LITERALLY 303 380 393 399 422 427 470 471
220			DN . . . . . LITERALLY 279 376 414 433 468 475
223	0050H	5	DSM. . . . . BYTE ARRAY(5) DATA 425
222	0028H	10	DSMAX. . . . . WORD ARRAY(5) DATA 307
79	0000H		EFINIS . . . . . PROCEDURE EXTERNAL(29) STACK=0000H 579
154	0145H	53	ELEMENT. . . . . PROCEDURE BYTE STACK=0004H 543

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER #

```
153  0026H    1  ELTINX . . . . . BYTE       157  159  546
 40  0000H       EMITADDR . . . . . PROCEDURE EXTERNAL(15) STACK=0000H       202  205  207
 30  0000H       EMITBYTE . . . . . PROCEDURE EXTERNAL(11) STACK=0000H       176  193  198  250  253  34
                                     373  381  386  488  499  508  514
 43  0000H       EMITCHAR . . . . . PROCEDURE EXTERNAL(16) STACK=0000H       175  189  210  255  349  35
                                     375  377  385  388  490  493  497  520  524  527
 48  0000H       EMITCHARN. . . . . PROCEDURE EXTERNAL(18) STACK=0000H       196  200  211  214  216  24
                                     245  414  433  468  475  506  510  512  516
 38  0000H       EMITCOMMA. . . . . PROCEDURE EXTERNAL(14) STACK=0000H       201  206  215  448
 51  0000H       EMITCOMMENT. . . . PROCEDURE EXTERNAL(19) STACK=0000H       194  203  208  212  217  24
                                     417  438  442  474  478  496  502  511  517  523  528
 33  0000H       EMITCRLF . . . . . PROCEDURE EXTERNAL(12) STACK=0000H       256  353  378  389  451  46
                                     491
 65  0000H       EMITDB . . . . . PROCEDURE EXTERNAL(24) STACK=0000H       453
 70  0000H       EMITDBN. . . . . PROCEDURE EXTERNAL(26) STACK=0000H
 76  0000H       EMITDBNC . . . . . PROCEDURE EXTERNAL(28) STACK=0000H       419  420  425  426  429  43
                                     529
 57  0000H       EMITDEC16. . . . . PROCEDURE EXTERNAL(21) STACK=0000H       456  473  477
 54  0000H       EMITDEC8 . . . . . PROCEDURE EXTERNAL(20) STACK=0000H       350  352  376  437  501
 60  0000H       EMITDECB . . . . . PROCEDURE EXTERNAL(22) STACK=0000H       252  384
173  01A0H   27  EMITDOLLAR . . . . PROCEDURE STACK=0006H         192  416  441  495  522  526
 63  0000H       EMITDW . . . . . PROCEDURE EXTERNAL(23) STACK=0000H       199  204  209  213  454
 67  0000H       EMITDWN. . . . . PROCEDURE EXTERNAL(25) STACK=0000H
 73  0000H       EMITDWNC . . . . . PROCEDURE EXTERNAL(27) STACK=0000H       418  427  428  431  432
 46  0000H       EMITLINE . . . . . PROCEDURE EXTERNAL(17) STACK=0000H       540
 83  0000H       EMITNULCOM . . . . PROCEDURE EXTERNAL(31) STACK=0000H       346  354  370  412  487  492
 35  0000H       EMITOPTAB. . . . . PROCEDURE EXTERNAL(13) STACK=0000H       190  244  415  434  469  476
                                     484  485  494  521  525
 81  0000H       EMITTAB. . . . . PROCEDURE EXTERNAL(30) STACK=0000H       251  254  348  374  382  387
                                     489
480  097FH   272 ENDEF. . . . . PROCEDURE STACK=000AH          564
165  017AH    19 ENDLINE. . . . . PROCEDURE BYTE STACK=0002H          169  263  338
 86  0000H     1 EOFSET . . . . . BYTE EXTERNAL(34)          140  166  537
241  092FH    40 EQUIV. . . . . PROCEDURE STACK=000CH          356  357  358  359
 27  0000H       ERRPTR . . . . . PROCEDURE EXTERNAL(10) STACK=0000H       187  226  267  271  275  282
                                     321  343  364  544  568
 86  0000H     1 ERRSET . . . . . BYTE EXTERNAL(33)          334  538
  3             FALSE. . . . . LITERALLY        151  160  221  534  538
  3             FOREVER. . . . . LITERALLY        158  233
220             FSC. . . . . LITERALLY        285  287  367  456
231  0905H    42 GCD. . . . . PROCEDURE WORD STACK=0006H          444
  1  0000H       GD . . . . . PROCEDURE STACK=0000H
 96  0004H     1 H. . . . . BYTE PARAMETER AUTOMATIC          97   98   99
                 HIGH . . . . . BUILTIN        103  317  429
481  0046H     1 I. . . . . BYTE        503  504  506  510  512  516
532  0047H     1 I. . . . . BYTE        533  534
229  0045H     1 I. . . . . BYTE        258  259  421  423  425
118  0024H     1 I. . . . . BYTE        124  125
180  0039H     1 I. . . . . BYTE        195  196  200  211  214  216
116             IDENT. . . . . LITERALLY
                LAST . . . . . BUILTIN        258  503  533
154  0004H     2 LIST . . . . . WORD PARAMETER AUTOMATIC          155  156  159
  2             LIT. . . . . LITERALLY          2    3  116  178  220
                LOW. . . . . BUILTIN        104  319  430
221  0000H    20 LOWER. . . . . WORD ARRAY(10) DATA          277
220             LSC. . . . . LITERALLY        285  287  367
231  0006H     2 M. . . . . WORD PARAMETER AUTOMATIC          232  234  237
```

COPYRIGHT © 1981,
DIGITAL RESEARC
P. O. BOX 579
PACIFIC GROVE, CA 9
SER. #

```
178                MAXDISKS . . . . .  LITERALLY       178  186
 27  0000H     2   MSG. . . . . .  WORD PARAMETER         28
 14  0000H     2   MSG. . . . . .  WORD PARAMETER         15
117  0004H     2   MSGA . . . . . .  WORD PARAMETER AUTOMATIC       118  122
 70  0000H     1   N. . . . . .  BYTE PARAMETER         71
 67  0000H     2   N. . . . . .  WORD PARAMETER         68
 48  0000H     1   N. . . . . .  BYTE PARAMETER         49
231  0004H     2   N. . . . . .  WORD PARAMETER AUTOMATIC       232  234  236  237  238
 76  0000H     1   N. . . . . .  BYTE PARAMETER         77
 73  0000H     2   N. . . . . .  WORD PARAMETER         74
178  0028H     1   NDISKS . . . . . .  BYTE        181  182  184  185  186  195  536
230  0020H     2   NELTS. . . . . .  WORD        445  460  463
230  001EH     2   NELTST . . . . . .  WORD        444  445  463
 86  0000H     1   NEXTC. . . . . .  BYTE EXTERNAL(37)
220                NPAR . . . . . .  LITERALLY       229  263  362
116                NUMBER . . . . . .  LITERALLY       134  183  274
230  001CH     2   NXTBAS . . . . . .  WORD        443  462
230  001AH     2   NXTSEC . . . . . .  WORD        443  456  457  458  459  462
 85  000EH     1   OFFTOG . . . . . .  BYTE EXTERNAL(32) AT       191
220                OFS. . . . . .  LITERALLY       407  432
221  003CH    10   OPTIONAL . . . . . .  BYTE ARRAY(10) DATA        270
 85  0000H    26   PARM . . . . . .  BYTE ARRAY(26) EXTERNAL(32)          85
229  003AH    10   PARMISSING . . . .  BYTE ARRAY(10)        258  259  268  371  396  409  410  424  435
229  0044H     1   PC . . . . . .  BYTE        261  263  265  268  270  277  279  285  287  289  303  310
                                      328  330  336  340  362
573  0B67H    26   PLM. . . . . .  PROCEDURE PUBLIC STACK=0026H
101  003EH    23   PRADDR . . . . . .  PROCEDURE STACK=0014H          137
110  0065H    59   PRDEC. . . . . .  PROCEDURE STACK=000EH          133
106  0055H    16   PRDIG. . . . . .  PROCEDURE STACK=0008H          112  113  114
 96  0023H    27   PRHEX. . . . . .  PROCEDURE STACK=000EH          103  104  128
  9  0000H        PRINT. . . . . .  PROCEDURE EXTERNAL(2) STACK=0000H        122  141
  6  0000H        PRINTCHAR. . . . .  PROCEDURE EXTERNAL(1) STACK=0000H         94  108  123  127  130  132
                                      136  138
 89  0000H    35   PRNIB. . . . . .  PROCEDURE STACK=0008H           98   99
531  0AA7H   192   PROGRAM. . . . . .  PROCEDURE STACK=0022H          577
 23  0000H        PUTLINE. . . . . .  PROCEDURE EXTERNAL(8) STACK=0000H        121
229  0000H    20   PV . . . . . .  WORD ARRAY(10)        243  245  265  285  287  305  307  318  328  330
                                      342  350  352  367  376  380  391  392  393  394  395  399  402  405
                                      407  408  411  414  419  420  421  422  426  427  428  431  432  433
                                      444  456  457  468  470  471  475  477
232  0022H     2   R. . . . . .  WORD        234  235  238
225  02CCH    12   RANGE. . . . . .  PROCEDURE STACK=0006H          278  286  288  301  306  308  313  329
168  018DH    19   RECOVER. . . . . .  PROCEDURE STACK=0006H          570
482  0A8FH    24   RESERVE. . . . . .  PROCEDURE STACK=0006H          500  509  515
                   ROL. . . . . .  BUILTIN         317
 17  0000H        SCAN . . . . . .  PROCEDURE EXTERNAL(5) STACK=0000H        148  170  188  332  539  548
                                      550  554  563
 19  0000H        SCANINI. . . . . .  PROCEDURE EXTERNAL(6) STACK=0000H        575
 85  0012H     1   SCANTOG. . . . . .  BYTE EXTERNAL(32) AT       119
230  0018H     2   SECTORS. . . . . .  WORD        367  406  418  444  446  452  458  459
156  0000H     1   SET. . . . . .  BYTE BASED(LIST) ARRAY(1)       159
 87  0000H        SETEMIT. . . . . .  PROCEDURE EXTERNAL(42) STACK=0000H        574
 21  0000H        SETUP. . . . . .  PROCEDURE EXTERNAL(7) STACK=0000H        576
                   SHL. . . . . .  BUILTIN         380  392  402  405
                   SHR. . . . . .  BUILTIN          98  324
220                SKF. . . . . .  LITERALLY       287  408  409  410  411  435  444  457
116                SPECIAL. . . . . .  LITERALLY
```

COPYRIGHT © 1981, 198
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 9395
SER. #

```
   116                  STRING . . . . . .  LITERALLY
     3                  TAB. . . . . .  LITERALLY
     2                  TDISKDEF . . . . .  LITERALLY       543
     2                  TDISKS . . . . . .  LITERALLY       543
     2                  TENDEF . . . . . .  LITERALLY       543
     2                  TIDEN. . . . . .  LITERALLY
     2                  TMACLIB. . . . . .  LITERALLY       543
     2                  TNUMB. . . . . .  LITERALLY
    86  0000H     1     TOKEN. . . . . .  BYTE EXTERNAL(38)        133  134  146  161  166  183  262  268  27
                                           542  567
     2                  TOKEN2 . . . . . .  LITERALLY
     2                  TOKEN3 . . . . . .  LITERALLY
     2                  TOKEN4 . . . . . .  LITERALLY
     2                  TOKEN5 . . . . . .  LITERALLY
     2                  TOKEN6 . . . . . .  LITERALLY
     2                  TOKEN7 . . . . . .  LITERALLY
     2                  TOKEN8 . . . . . .  LITERALLY
     2                  TOKENM . . . . . .  LITERALLY
   117  00A0H   141     TRACE. . . . . .  PROCEDURE STACK=001AH         264  541  549  551  555  559
     3                  TRUE . . . . . .  LITERALLY       149  158  162  221  233  259  283  409
     2                  TSPECL . . . . . .  LITERALLY
     2                  TSTRNG . . . . . .  LITERALLY
   221  0014H    20     UPPER. . . . . .  WORD ARRAY(10) DATA       277
   248  0006H     2     V. . . . . .  WORD PARAMETER AUTOMATIC       249  252
   229  0016H     2     V. . . . . .  WORD       315  319  324  429  430
    86  0000H     2     VALUE. . . . . .  WORD EXTERNAL(36)        137  184  277  281  283  285  287  291  292
                                           293  294  295  296  297  298  299  300  305  307  312  316  328  330
    60  0000H     2     W. . . . . .  WORD PARAMETER        61
    57  0000H     2     W. . . . . .  WORD PARAMETER        58
    25  0000H           WRITERRS . . . . .  PROCEDURE EXTERNAL(9) STACK=0000H        578
   156  0027H     1     X. . . . . .  BYTE       159  161
   144  0004H     1     X. . . . . .  BYTE PARAMETER AUTOMATIC       145  146
    85  0019H     1     Z80TOG . . . . . .  BYTE EXTERNAL(32) AT     174  197  483  498  507  513
```

MODULE INFORMATION:

```
     CODE AREA SIZE     = 0B81H   2945D
     CONSTANT AREA SIZE = 04EEH   1262D
     VARIABLE AREA SIZE = 0048H     72D
     MAXIMUM STACK SIZE = 0026H     38D
     823 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

CP/M-86 v.1.1 (Vol. II)
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # C81-162

```
ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE EMIT
OBJECT MODULE PLACED IN EMIT.OBJ
COMPILER INVOKED BY:  :F0:EMIT.PLM DEBUG XREF OPTIMIZE(2) DATE(APRIL 9, 1982)
```

```
  1          emit:
             do;
             /* D I S K D E F   l i b   g e n e r a t o r   e m i t t e r */

  2   1      mon1: procedure(f,a) external;
  3   2          declare f byte, a address;
  4   2          end mon1;

  5   1      mon2: procedure(f,a) byte external;
  6   2          declare f byte, a address;
  7   2          end mon2;

  8   1      boot: procedure public;
  9   2          call mon1(0,0);
 10   2          end boot;

 11   1      abort: procedure(msg) external;
 12   2          declare msg address;
 13   2          end abort;

             /* global variables */
 14   1      declare
             parm (26) byte external;  /* initial parameters */
             /* a b c d e f g h i j k l m n o p q r s t u v w x y z
                0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2
                0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 */

 15   1      declare
             acclen   byte external,    /* accumulator length */
             accum(32)byte external;    /* accumulator */

 16   1      declare
             lbuff(75) byte external,   /* physical line */
             lblen     byte external,   /* lbuff length */
             lineready byte external,   /* line ready flag */
             lineout   byte external,   /* true if current line out */
             cbuff(8)  byte external;   /* line prefix */

 17   1      declare
             lit literally 'literally',
             true    lit '1',
             false   lit '0',
             forever lit 'while true',
             eofile  lit '1ah',         /* end of file */
             tab     lit '09h',         /* horizontal tab */
             cr      lit '0dh',
             lf      lit '0ah';

 18   1      declare dcnt byte external;

 19   1      close: procedure(fcb) external;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
20  2       declare fcb address;
21  2       end close;

22  1   delete: procedure(fcb) external;
23  2       declare fcb address;
24  2       end delete;

25  1   diskwrite: procedure(fcb) byte external;
26  2       declare fcb address;
27  2       end diskwrite;

28  1   make: procedure(fcb) external;
29  2       declare fcb address;
30  2       end make;

31  1   rename: procedure(fcb) external;
32  2       declare fcb address;
33  2       end rename;

34  1   setdma: procedure(dma) external;
35  2       declare dma address;
36  2       end setdma;

        /* local variables */
37  1   declare
            ifcb (33) byte external,  /* initial input file control block */
            efcb (33) byte,           /* .asm file control block */
            eufb  byte at (.efcb(13)),/* unfilled bytes field */
            efcbr byte at (.efcb(32));/* next record field */

        /* buffers */
38  1   declare
            comcol lit '32',          /* beginning column for comments */
            ecolumn    byte,          /* output column 0,1,... */
            ebuff (512) byte,         /* emit file buffer */
            ebp address;              /* emit buffer pointer */

        /* sector size definitions */
39  1   declare
            sectsize lit '128',       /* bytes per sector */
            sectmsk  lit '7fh',       /* sectsize - 1 */
            sectshf  lit '7';         /* shl(1,sectshf) = sectsize */

40  1   setemit: procedure public;
            /* set up code emitter file */
41  2       call move(16,.ifcb(0),.efcb(0)); /* copy name */
42  2       call move(4,.('LIB',0),.efcb(9));/* copy type */
43  2       if ifcb(16) <> 0 then /* select destination drive */
44  2           efcb(0) = ifcb(16);
45  2       call delete(.efcb);
46  2       call make(.efcb);
47  2       if dcnt = 255 then /* no space */
48  2           call abort(.('no ".LIB" directory space$'));
49  2       efcbr,ebp,ecolumn = 0;
50  2       lineout = true; /* prevent initial line write */
51  2       end setemit;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
52   1      clearebuff: procedure;
                /* clear the ebuff buffer up to (not including) ebp */
53   2          declare (i,n) byte;
                /* skip the case of no buffers to write */
54   2          if low(n:=shr(ebp,7)-1) = 255 then return;
                /* n is last buffer to write */
56   2          ebp = 0;
57   2              do i = 0 to n;
58   3              call setdma(.ebuff(ebp));
59   3              if diskwrite(.efcb) <> 0 then
60   3                  call abort(.('".LIB" disk full$'));
61   3              ebp = ebp + sectsize;
62   3              end;
63   2          ebp = 0;
64   2          end clearebuff;

65   1      emitbyte: procedure(b) public;
66   2          declare b byte;
                /* write b to emit file */
67   2          if b = cr then
68   2              ecolumn = 0;
69   2          if b = tab then
70   2              ecolumn = (ecolumn + 8) and 11111000b;
71   2          if b >= ' ' then
72   2              ecolumn = ecolumn + 1;
73   2          if ebp >= length(ebuff) then call clearebuff;
75   2          ebuff(ebp) = b;
76   2          ebp = ebp + 1;
77   2          end emitbyte;

78   1      efinis: procedure public;
                /* emit file finis */
79   2          eufb = 0; /* clear the unfilled bytes field */
80   2              do while (low(ebp) and sectmsk) <> 0;
81   3              eufb = eufb + 1; /* counts unfilled bytes */
82   3              call emitbyte(eofile);
83   3              end;
84   2          call clearebuff; /* write all buffers */
85   2          call close(.efcb);
86   2          if dcnt = 255 then
87   2              call abort(.('cannot close ".LIB"$'));
88   2          end efinis;

89   1      emitcrlf: procedure public;
                /* emit end of physical line */
90   2          call emitbyte(cr);
91   2          call emitbyte(lf);
92   2          end emitcrlf;

93   1      emittab: procedure public;
94   2          call emitbyte(tab);
95   2          end emittab;

96   1      emitcomma: procedure public;
97   2          call emitbyte(',');
98   2          end emitcomma;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
 99  1     emitdigit: procedure(d);
100  2         declare d byte;
                 /* emit the decimal digit given by d */
101  2         call emitbyte('0'+d);
102  2         end emitdigit;

103  1     emitnulcom: procedure public;
104  2         call emitbyte(';');
105  2         call emitcrlf;
106  2         end emitnulcom;

107  1     declare
             /* literals for decimal output control */
             zsup$on   lit 'true',   /* zero suppression on */
             zsup$off  lit 'false',  /* zero suppression off */
             bsup$on   lit 'true',   /* blank suppress */
             bsup$off  lit 'false',  /* blank suppress off */
             byte$base lit '100',    /* byte value */
             word$base lit '10000';  /* word value */

108  1     emitdecz: procedure(v,zsup,bsup,base) public;
109  2         declare
                 v address, /* value to emit */
                 zsup byte, /* zero suppress if true */
                 bsup byte,    /* blank suppression */
                 base address, /* 100 for byte, 10000 for word */
                 d byte;
110  2         do while base <> 0;
111  3         d = v/base mod 10;
112  3         if (zsup := zsup and (d = 0) and (base > 1)) then
113  3             do;
114  4             if not bsup then call emitbyte(' ');
116  4             end; else
117  3             call emitdigit(d);
118  3         base = base/10;
119  3         end;
120  2         end emitdecz;

121  1     emitnib: procedure(n);
122  2         declare n byte;
123  2         if n > 9 then
124  2             n = n - 10 + 'A'; else
125  2             n = n + '0';
126  2         call emitbyte(n);
127  2         end emitnib;

128  1     emithex8: procedure(h) public;
129  2         declare h byte;
130  2         call emitnib(shr(h,4));
131  2         call emitnib(h and 0fh);
132  2         end emithex8;

133  1     emithex16: procedure(a) public;
134  2         declare a address;
135  2         call emithex8(high(a));
136  2         call emithex8(low(a));
137  2         end emithex16;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
138  1      emitaddr: procedure(a) public;
139  2          declare a address;
140  2          call emithex16(a);
141  2          call emitbyte('h');
142  2          end emitaddr;

143  1      emitdec8: procedure(b) public;
144  2          declare b byte;
145  2          call emitdecz(b,zsup$on,bsup$on,byte$base);
146  2          end emitdec8;

147  1      emitdec16: procedure(w) public;
148  2          declare w address;
149  2          call emitdecz(w,zsup$on,bsup$on,word$base);
150  2          end emitdec16;

151  1      emitdecb: procedure(w) public;
152  2          declare w address;
153  2          call emitdecz(w,zsup$on,bsup$off,word$base);
154  2          end emitdecb;

155  1      emitline: procedure public;
                /* write current line to emit file */
156  2          declare i byte;
157  2          if lineready then /* line is ready to write */
158  2              do;
159  3              if lineout then return; /* already written */
161  3              lineout = true;
162  3              if lblen = 0 then return;
                    /* non - zero line length */
164  3              call emitbyte(';');
165  3              call emittab;
166  3              i = 0;
167  3                  do while (i:=i+1) < lblen;
168  4                  call emitbyte(lbuff(i-1));
169  4                  end;
170  3              call emitcrlf;
171  3              end;
172  2          end emitline;

173  1      emitchar: procedure(a) public;
                /* emit literal string */
174  2          declare a address;
175  2          declare s based a byte;
176  2              do while s <> '$';
177  3              call emitbyte(s);
178  3              a = a + 1;
179  3              end;
180  2          end emitchar;

181  1      emitcharn: procedure(a,n) public;
182  2          declare
                    a address,
                    n byte;
                /* emitchar(a), followed by number */
183  2          call emitchar(a);
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
184  2          call emitdecz(n,zsup$on,bsup$on,byte$base);
185  2          end emitchorn;

186  1      emitop: procedure(a) public;
                /* tab, operator */
187  2          declare a address;
188  2          call emittab;
189  2          call emitchar(a);
190  2          end emitop;

191  1      emitoptab: procedure(a) public;
                /* emit tab, operator, tab */
192  2          declare a address;
193  2          call emitop(a);
194  2          call emittab;
195  2          end emitoptab;

196  1      emitcomment: procedure(a) public;
197  2          declare a address;
198  2              do while ecolumn < comcol;
199  3              call emittab;
200  3              end;
201  2          call emitbyte(';');
202  2          call emitchar(a);
203  2          call emitcrlf;
204  2          end emitcomment;

205  1      emitdw: procedure public;
206  2          call emitoptab(.('dw$'));
207  2          end emitdw;

208  1      emitdb: procedure public;
209  2          call emitoptab(.('db$'));
210  2          end emitdb;

211  1      emitdwn: procedure(n) public;
212  2          declare n address;
213  2          call emitdw;
214  2          call emitdec16(n);
215  2          end emitdwn;

216  1      emitdbn: procedure(n) public;
217  2          declare n byte;
218  2          call emitdb;
219  2          call emitdec8(n);
220  2          end emitdbn;

221  1      emitdwnc: procedure(n,c) public;
222  2          declare (n,c) address;
223  2          call emitdwn(n);
224  2          call emitcomment(c);
225  2          end emitdwnc;

226  1      emitdbnc: procedure(n,c) public;
227  2          declare n byte, c address;
228  2          call emitdbn(n);
229  2          call emitcomment(c);
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
230  2        end emitdbnc;

231  1    end;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
133	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 134 135 136
5	0000H	2	A. . . . . WORD PARAMETER 6
181	0006H	2	A. . . . . WORD PARAMETER AUTOMATIC 182 183
186	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 187 189
196	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 197 202
2	0000H	2	A. . . . . WORD PARAMETER 3
173	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 174 175 176 177 178
138	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 139 140
191	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 192 193
11	0000H		ABORT. . . . . PROCEDURE EXTERNAL(2) STACK=0000H 48 60 87
15	0000H	1	ACCLEN . . . . . BYTE EXTERNAL(4)
15	0000H	32	ACCUM. . . . . BYTE ARRAY(32) EXTERNAL(5)
143	0004H	1	B. . . . . BYTE PARAMETER AUTOMATIC 144 145
65	0004H	1	B. . . . . BYTE PARAMETER AUTOMATIC 66 67 69 71 75
108	0004H	2	BASE . . . . . WORD PARAMETER AUTOMATIC 109 110 111 112 118
8	0000H	14	BOOT . . . . . PROCEDURE PUBLIC STACK=0008H
108	0006H	1	BSUP . . . . . BYTE PARAMETER AUTOMATIC 109 114
107			BSUPOFF. . . . . LITERALLY 153
107			BSUPON . . . . . LITERALLY 145 149 184
107			BYTEBASE . . . . . LITERALLY 145 184
226	0004H	2	C. . . . . WORD PARAMETER AUTOMATIC 227 229
221	0004H	2	C. . . . . WORD PARAMETER AUTOMATIC 222 224
16	0000H	8	CBUFF. . . . . BYTE ARRAY(8) EXTERNAL(10)
52	006AH	90	CLEAREBUFF . . . . PROCEDURE STACK=0006H 74 84
17	0000H		CLOSE. . . . . PROCEDURE EXTERNAL(12) STACK=0000H 85
38			COMCOL . . . . . LITERALLY 198
17			CR . . . . . LITERALLY 87 90
99	0004H	1	D. . . . . BYTE PARAMETER AUTOMATIC 100 101
109	0226H	1	D. . . . . BYTE 111 112 117
18	0000H	1	DCNT . . . . . BYTE EXTERNAL(11) 47 86
22	0000H		DELETE . . . . . PROCEDURE EXTERNAL(13) STACK=0000H 45
25	0000H		DISKWRITE. . . . PROCEDURE BYTE EXTERNAL(14) STACK=0000H 59
34	0000H	2	DMA. . . . . WORD PARAMETER 35
38	0000H	2	EBP. . . . . WORD 49 54 56 58 61 63 73 75 76 80
38	0024H	512	EBUFF. . . . . BYTE ARRAY(512) 58 73 75
38	0023H	1	ECOLUMN. . . . . BYTE 49 68 70 72 198
37	0002H	33	EFCB . . . . . BYTE ARRAY(33) 37 41 42 44 45 46 59 85
37	0022H	1	EFCBR. . . . . BYTE AT 49
78	010AH	53	EFINIS . . . . . PROCEDURE PUBLIC STACK=0010H
1	0000H		EMIT . . . . . PROCEDURE STACK=0000H
138	0242H	19	EMITADDR . . . . PROCEDURE PUBLIC STACK=0024H
65	00C4H	70	EMITBYTE . . . . PROCEDURE PUBLIC STACK=000CH 82 90 91 94 97 101 104 115 126 141 164 168 177 201
173	02EAH	25	EMITCHAR . . . . PROCEDURE PUBLIC STACK=0012H 183 189 202
181	0303H	30	EMITCHARN. . . . PROCEDURE PUBLIC STACK=0026H
96	015BH	11	EMITCOMMA. . . . PROCEDURE PUBLIC STACK=0010H
176	0341H	34	EMITCOMMENT. . . PROCEDURE PUBLIC STACK=0018H 224 229
89	013FH	17	EMITCRLF . . . . PROCEDURE PUBLIC STACK=0010H 105 170 203
208	036FH	12	EMITDB . . . . . PROCEDURE PUBLIC STACK=0022H 218

DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 9395

```
216  0388H   16  EMITDBN. . . . . PROCEDURE PUBLIC STACK=002AH        228
226  03AEH   19  EMITDBNC . . . . . PROCEDURE PUBLIC STACK=0032H
147  026DH   21  EMITDEC16. . . . . PROCEDURE PUBLIC STACK=0024H        214
143  0255H   24  EMITDEC8 . . . . . PROCEDURE PUBLIC STACK=0024H        219
151  0282H   23  EMITDECB . . . . . PROCEDURE PUBLIC STACK=0024H
108  0184H  105  EMITDECZ . . . . . PROCEDURE PUBLIC STACK=001EH        145  149  153  184
 99  0166H   16  EMITDIGIT. . . . . PROCEDURE STACK=0012H          117
205  0363H   12  EMITDW . . . . . PROCEDURE PUBLIC STACK=0022H        213
211  037BH   16  EMITDWN. . . . . PROCEDURE PUBLIC STACK=002AH.       223
221  039BH   19  EMITDWNC . . . . . PROCEDURE PUBLIC STACK=0032H
133  022BH   23  EMITHEX16. . . .  PROCEDURE PUBLIC STACK=001EH        140
128  0210H   27  EMITHEX8 . . . . . PROCEDURE PUBLIC STACK=0018H        135  136
155  0299H   81  EMITLINE . . . . . PROCEDURE PUBLIC STACK=0014H
121  01EDH   35  EMITNIB. . . . . PROCEDURE STACK=0012H          130  131
103  0176H   14  EMITNULCOM . . . . PROCEDURE PUBLIC STACK=0014H
186  0321H   16  EMITOP . . . . . PROCEDURE PUBLIC STACK=0018H        193
191  0331H   16  EMITOPTAB. . . . . PROCEDURE PUBLIC STACK=001EH        206  209
 93  0150H   11  EMITTAB. . . . . PROCEDURE PUBLIC STACK=0010H        165  188  194  199
 17          EOFILE . . . . . LITERALLY        82
 37  000FH    1  EUFB . . . . . . . BYTE AT          79   81
  2  0000H    1  F. . . . . . . . . BYTE PARAMETER          3
  5  0000H    1  F. . . . . . . . . BYTE PARAMETER          6
 17          FALSE. . . . . LITERALLY       153
 31  0000H    2  FCB. . . . . . . . WORD PARAMETER         32
 28  0000H    2  FCB. . . . . . . . WORD PARAMETER         29
 25  0000H    2  FCB. . . . . . . . WORD PARAMETER         26
 22  0000H    2  FCB. . . . . . . . WORD PARAMETER         23
 19  0000H    2  FCB. . . . . . . . WORD PARAMETER         20
 17          FOREVER. . . . . LITERALLY
128  0004H    1  H. . . . . . . . . BYTE PARAMETER AUTOMATIC    129  130  131
          HIGH . . . . . BUILTIN         135
 53  0224H    1  I. . . . . . . . . BYTE        57
156  0227H    1  I. . . . . . . . . BYTE       166  167  168
 37  0000H   33  IFCB . . . . . . . BYTE ARRAY(33) EXTERNAL(18)          41   43   44
 16  0000H    1  LBLEN. . . . . . . BYTE EXTERNAL(7)          162  167
 16  0000H   75  LBUFF. . . . . . . BYTE ARRAY(75) EXTERNAL(6)          168
          LENGTH . . . . . BUILTIN         73
 17          LF . . . . . . . . LITERALLY        91
 16  0000H    1  LINEOUT. . . . . . BYTE EXTERNAL(9)           50  159  161
 16  0000H    1  LINEREADY. . . . . BYTE EXTERNAL(8)          157
 17          LIT. . . . . . . . LITERALLY        17   38   39  107
          LOW. . . . . . . . BUILTIN         54   80  136
 28  0000H        MAKE . . . . . . . PROCEDURE EXTERNAL(15) STACK=0000H         46
  2  0000H        MON1 . . . . . . . PROCEDURE EXTERNAL(0) STACK=0000H          9
  5  0000H        MON2 . . . . . . . PROCEDURE BYTE EXTERNAL(1) STACK=0000H
          MOVE . . . . . . . BUILTIN         41   42
 11  0000H    2  MSG. . . . . . . . WORD PARAMETER         12
 53  0225H    1  N. . . . . . . . . BYTE        54   57
226  0006H    1  N. . . . . . . . . BYTE PARAMETER AUTOMATIC       227  228
221  0006H    2  N. . . . . . . . . WORD PARAMETER AUTOMATIC       222  223
216  0004H    1  N. . . . . . . . . BYTE PARAMETER AUTOMATIC       217  219
211  0004H    2  N. . . . . . . . . WORD PARAMETER AUTOMATIC       212  214
181  0004H    1  N. . . . . . . . . BYTE PARAMETER AUTOMATIC       182  184
121  0004H    1  N. . . . . . . . . BYTE PARAMETER AUTOMATIC       122  123  124  125  126
 14  0000H   26  PARM . . . . . . . BYTE ARRAY(26) EXTERNAL(3)
 31  0000H       RENAME . . . . . PROCEDURE EXTERNAL(16) STACK=0000H
175  0000H    1  S. . . . . . . . . BYTE BASED(A)        176  177
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

```
 39                 SECTMSK. . . . . .  LITERALLY       80
 39                 SECTSHF. . . . . .  LITERALLY
 39                 SECTSIZE . . . . .  LITERALLY       61
 34  0000H          SETDMA . . . . . .  PROCEDURE EXTERNAL(17) STACK=0000H        58
 40  000EH    92    SETEMIT. . . . . .  PROCEDURE PUBLIC STACK=0006H
                    SHR. . . . . . . .  BUILTIN         54  130
 17                 TAB. . . . . . . .  LITERALLY       69   94
 17                 TRUE . . . . . . .  LITERALLY       50  145  149  153  161  184
108  000AH     2    V. . . . . . . . .  WORD PARAMETER AUTOMATIC      109  111
151  0004H     2    W. . . . . . . . .  WORD PARAMETER AUTOMATIC      152  153
147  0004H     2    W. . . . . . . . .  WORD PARAMETER AUTOMATIC      148  149
107                 WORDBASE . . . . .  LITERALLY       149  153
108  0008H     1    ZSUP . . . . . . .  BYTE PARAMETER AUTOMATIC      109  112
107                 ZSUPOFF. . . . . .  LITERALLY
107                 ZSUPON . . . . . .  LITERALLY       145  149  153  184
```

MODULE INFORMATION:

```
    CODE AREA SIZE     = 03C1H    961D
    CONSTANT AREA SIZE = 0049H     73D
    VARIABLE AREA SIZE = 0228H    552D
    MAXIMUM STACK SIZE = 0032H     50D
    339 LINES READ
    0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE SCANMOD
OBJECT MODULE PLACED IN GSCAN.OBJ
COMPILER INVOKED BY:  :F0:GSCAN.PLM OPTIMIZE(2) DEBUG XREF DATE( 1/29/82 )


  1          scanmod:
             do;
             /* D I S K D E F   l i b   s c a n n e r   m o d u l e */
  2   1      declare
                 major literally '''1''',   /* major release number */
                 minor literally '''0''';   /* minor release number */

  3   1      declare
                 parm (26) byte public;    /* initial parameters */
                 /* a b c d e f g h i j k l m n o p q r s t u v w x y z
                    0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 2 2 2 2 2 2
                    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 */

  4   1      declare
                 defbuf literally '0080h';  /* default buffer */

  5   1      mon1: procedure(f,a) external;
  6   2          declare f byte, a address;
  7   2          end mon1;

  8   1      mon2: procedure(f,a) byte external;
  9   2          declare f byte, a address;
 10   2          end mon2;

 11   1      boot: procedure external;
 12   2          end boot;

             $include (:f1:pas.plb)
 13   1  =   declare
        =        true literally '1',
        =        false literally '0',
        =        forever literally 'while true',
        =        cr literally '13',
        =        lf literally '10';
        =
 14   1  =   printchar: procedure(char) public;
 15   2  =       declare char byte;
 16   2  =       call mon1(2,char);
 17   2  =       end printchar;
        =
 18   1  =   crlf: procedure public;
 19   2  =       call printchar(cr);
 20   2  =       call printchar(lf);
 21   2  =       end crlf;
        =
 22   1  =   printn: procedure(a) public;
 23   2  =       declare a address;
 24   2  =       call mon1(9,a);
 25   2  =       end printn;
        =
 26   1  =   print: procedure(a) public;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
27  2  =      declare a address;
       =      /* print the string starting at address a until the
       =      next dollar sign is encountered */
28  2  =      call crlf;
29  2  =      call printn(a);
30  2  =      end print;
       =
31  1  =  declare dcnt byte public;
       =
32  1  =  open: procedure(fcb) public;
33  2  =      declare fcb address;
34  2  =      dcnt = mon2(15,fcb);
35  2  =      end open;
       =
36  1  =  close: procedure(fcb) public;
37  2  =      declare fcb address;
38  2  =      dcnt = mon2(16,fcb);
39  2  =      end close;
       =
40  1  =  delete: procedure(fcb) public;
41  2  =      declare fcb address;
42  2  =      call mon1(19,fcb);
43  2  =      end delete;
       =
44  1  =  diskread: procedure(fcb) byte public;
45  2  =      declare fcb address;
46  2  =      return mon2(20,fcb);
47  2  =      end diskread;
       =
48  1  =  diskwrite: procedure(fcb) byte public;
49  2  =      declare fcb address;
50  2  =      return mon2(21,fcb);
51  2  =      end diskwrite;
       =
52  1  =  make: procedure(fcb) public;
53  2  =      declare fcb address;
54  2  =      dcnt = mon2(22,fcb);
55  2  =      end make;
       =
56  1  =  rename: procedure(fcb) public;
57  2  =      declare fcb address;
58  2  =      call mon1(23,fcb);
59  2  =      end rename;
       =
60  1  =  setdma: procedure(dma) public;
61  2  =      declare dma address;
62  2  =      call mon1(26,dma);
63  2  =      end setdma;

           /* file control blocks */
64  1      declare
           parms (8) byte  at (06dh),     /* $ parameters */
           ifcb (33) byte external,       /* default input fcb */
           ifcbr byte at (.ifcb(32));     /* record to read next */


           /* buffers */
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
65  1     declare
              dbuff (512) byte,          /* disk input buffer 128 * 4 */
              dbp address,               /* disk buffer pointer */
              lbuff(75) byte public,     /* 80 character line (5 char prefix) */
              lbp byte,                  /* line buffer pointer */
              lblen byte public,         /* line buffer length */
              lineout  byte public,      /* true if line out to .asm file */
              lineready byte public;     /* true if line is prepared for output */

66  1     declare
              errset byte public,        /* true if error already flagged */
              eofset byte public,        /* true if eof encountered */
              cbuff (8) byte public;


          /* miscellaneous non graphic characters */
67  1     declare
              lit literally 'literally',
              eofile   lit '1ah',    /* ascii end of file */
              eject    lit '0ch',    /* page eject for title */
              tab      lit '09h';    /* horizontal tab */

          /* code emitters for inline mode */
68  1     emitbyte: procedure(b) external;
69  2         declare b byte;
70  2         end emitbyte;

71  1     emitcrlf: procedure external;
72  2         end emitcrlf;

73  1     declare
              title (*) byte data
                 (eject,'   DISKDEF Table Generator, Vers ',major,'.',minor,
                  cr,lf,'$');

          /* error message */
74  1     declare
              /* the first positions are counted up in the errptr subroutine */
              errmsg (17) byte initial('       Error(s)$');

75  1     abort: procedure(msg) public;
              /* print message and reboot */
76  2         declare msg address;
77  2         call crlf;
78  2         call print(msg);
79  2         call boot;
80  2         end abort;

81  1     gnd: procedure byte;
              /* get next disk character */

82  2         checkeof: procedure byte;
                  /* check for end of file before returning */
83  3             declare c byte;
84  3             if (c := dbuff(dbp)) = eofile then eofset = true;
86  3             return c;
87  3             end checkeof;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
 88  2          if eofset then return eofile;
 90  2          if (dbp := dbp + 1) <= last(dbuff) then return checkeof;
                /* otherwise, read buffers to dbuff */
 92  2          dbp = 0;
 93  2              do while dbp < length(dbuff);
 94  3              call setdma(.dbuff(dbp));
 95  3              if diskread(.ifcb) <> 0 then /* end of file */
 96  3                  do; dbuff(dbp) = eofile; dbp = length(dbuff);
 99  4                  end;
                    else /* disk read was successful */
100  3                  dbp = dbp + 128;
101  3              end;
102  2          dbp = 0;
103  2          return checkeof;
104  2          end gnd;

105  1      incline: procedure(buffa);
106  2          declare buffa address;
                /* increment line number in buff */
107  2          declare buff based buffa (4) byte;
108  2          declare (i,c) byte;
109  2          i = 4;
110  2              do while (i := i - 1) <> 255;
111  3              if (c := buff(i)) = ' ' then c = '0';
113  3              if (buff(i) := c + 1) > '9' then
114  3                  buff(i) = '0'; else i = 0;
116  3              end;
117  2          end incline;

118  1      putline: procedure public;
                /* print the current line */
119  2          declare i byte;
120  2          if lineready then /* line has not yet been sent */
121  2              do;
122  3              call print(.cbuff); /* 7 character prefix */
123  3              if lblen > 0 then
124  3                  do i = 0 to lblen-1;
125  4                  call printchar(lbuff(i));
126  4                  end;
127  3              end;
128  2          lineready = false; /* marked as sent */
129  2          end putline;

130  1      getline: procedure;
                /* read next line and place into lbuff */
131  2          declare char byte;

132  2          putchar: procedure(c);
133  3              declare c byte;
134  3              if lbp+1 < length(lbuff) then
135  3                  do; lbuff(lbp:=lbp+1) = c;
137  4                  end;
138  3              end putchar;

                /* read line until overflow or lf or eofile */
139  2          call putline;
```

CP/M-86 v.1.1 (vol. II)
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # C81-162

```
140  2          lbp = -1; lblen = 0;
142  2          if (char := gnd) <> eofile then
143  2              do; call incline(.cbuff);
145  3                  do while not (char = lf or char = eofile);
146  4                  if char = cr then call putchar(cr); else
148  4                  if char = tab then /* expand to next tab position */
149  4                      do; call putchar(' ');
151  5                          do while ((lbp + 1) and 111b) <> 0;
152  6                          call putchar(' '); /* tabs at every 8 columns */
153  6                          end;
154  5                      end; else
155  4                  if char >= ' ' then /* graphic */
156  4                      do; /* convert lower to upper case alphabetics */
157  5                      if char <> 7fh then /* not a delete character */
158  5                          do; if (char - 'a') < 26 then /* lower alpha */
160  6                                  char = char and 5fh;  /* converted */
161  6                          call putchar(char); /* placed into line buffer */
162  6                          end;
163  5                      end;
                             char = gnd;
165  4                  end; /* end of file is detected in gnt */
166  3              lblen = lbp + 1;
167  3              lineready = true; /* not yet sent, but ready */
168  3              lineout = false;  /* not yet sent to .asm file */
169  3              end;
170  2          lbp = 0;
171  2          end getline;

172  1      setup: procedure public;
173  2          declare (i,c) byte;
               /* get the initial parameters */
174  2              do i = 0 to last(parm);
175  3              parm(i) = false;
176  3              end;
177  2          if parms(0) = '$' then
178  2              do i = 1 to 7;
179  3              if (c := parms(i) - 'A') < 26 then
180  3                  parm(c) = true;
181  3              end;
               /* set up the input file */
182  2          if ifcb(9) = ' ' then
183  2              call move(4,.('DEF',0),.ifcb(9));
184  2          call setdma(defbuf); /* reset to default area */
185  2          call open(.ifcb);
186  2          if dcnt = 255 then /* not present */
187  2              do;
188  3              call print(.(cr,lf,'No Input File Present, Command Form is:',
                                 cr,lf,cr,lf,
                                 'GENDEF x $'));
189  3              call printchar('$');
190  3              call printn(.('<parameters>',
                                 cr,lf,
                                 'Where x.DEF Holds Disk Definitions',
                                 cr,lf,
                                 'With Optional Parameters <parameters>:',
                                 cr,lf,
                                 'C:  Create Diskdef Comment',
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        cr,lf,
                        '0:  Generate Offset-Relative Labels',
                        cr,lf,
                        'Z:  Z80, 8080, or 8085 (else 8086, 8088)',
                        cr,lf,
                        'x.LIB is Created Upon Completion.',
                        cr,lf,'$'));
191   3             call boot;
192   3             end;
193   2         errset,eofset,lineready = false;
194   2         ifcbr = 0;  /* read starting at record 00 */
195   2         dbp = length(dbuff); /* causes immediate read from disk */
                /* initialize line and error counts */
196   2             do i=0 to last(cbuff);
197   3             cbuff(i) =  ' ';
198   3             errmsg(i) = ' ';
199   3             end;
200   2         cbuff(last(cbuff)) = '$';
                /* print title line */
201   2         call print(.title);
202   2         call getline; /* reads the first line */
203   2         end setup;

204   1     writerrs: procedure public;
                /* write error message at end of pass */
205   2         call putline;
206   2         if errmsg(6) = ' ' then
207   2             do; errmsg(5)='N'; errmsg(6)='o';
210   3             end;
211   2         call print(.errmsg(2));
212   2         end writerrs;

213   1     errptr: procedure(msg) public;
214   2         declare msg address;
                /* place question mark under error position */
215   2         declare i byte;
216   2         if errset then return;
                /* push line if not already printed */
218   2         call putline;
219   2         errset = true;

220   2         call incline(.errmsg(3)); /* count errors up */
221   2         call print(msg); /* message is seven characters long */
222   2         i = lbp - acclen;
223   2             do while (i := i - 1) > 0;
224   3             call printchar(' ');
225   3             end;
226   2         call printchar('?');
227   2         end errptr;

                /* literals for scanner tokens */
228   1     declare
                ident   lit '1',      /* identifier */
                number  lit '2',      /* number token */
                string  lit '3',      /* character string */
                special lit '4';      /* special character */
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
            /* scanner declarations */
229   1     declare
              value   address public, /* value for number token */
              nextc      byte public, /* next character look ahead symbol */
              token      byte public, /* vocabulary number for item scanned */
              continue   byte public, /* set if more of the same token remains */
              acclen     byte public, /* accumulator length */
              accum(32)  byte public;        /* accumulator */

230   1     putemitchar: procedure(b);
231   2         declare b byte;
232   2         if b <> 0 then call emitbyte(b);
234   2         end putemitchar;

235   1     gnc: procedure byte;
              /* get next input character, check for line boundaries */
236   2         declare c byte;
237   2             do while lbp >= lblen;
238   3             call getline;
239   3             if eofset then return eofile;
241   3             end;
              /* line read, character ready */
242   2         c = lbuff(lbp); lbp = lbp + 1;
244   2         return c;
245   2         end gnc;

246   1     declare
              errv lit '0',
              binv lit '2',
              octv lit '8',
              decv lit '10',
              hexv lit '16';

247   1     gnt: procedure public;
248   2         declare
                  (b, i, d, stype, lastc) byte,
                  v address;

249   2         numeric: procedure byte; return (nextc-'0') <= 9;
251   3             end numeric;

252   2         hex: procedure byte; return numeric or ((nextc-'A') <= 5);
254   3             end hex;

255   2         letter: procedure byte; return (nextc-'A') <= 25;
257   3             end letter;

258   2         alphanum: procedure byte; return numeric or letter;
260   3             end alphanum;

261   2         stype, acclen = 0;

262   2         if continue then
263   2             continue = false; else token = 0;

265   2             do while token = 0;
                  /* deblank input */
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
266  3          if nextc = ' ' or nextc = 0 then nextc = gnc; else
268  3          if letter then token = ident; else
270  3          if numeric then token = number; else
272  3          if nextc = '''' then
273  3              do; token = string; nextc = 0;
276  4              end; else
277  3          token = special;
278  3          if eofset then
279  3              call abort(.('Premature End-of-File$'));
280  3          end; /* of token = 0 */

                /* scan remainder of token */
281  2          do forever;
282  3          if nextc <> 0 then
283  3              do;
284  4              accum(acclen) = nextc;
285  4              if (acclen := acclen + 1) > last (accum) then
286  4                  do;
287  5                  if token = string then
288  5                      do;
289  6                      continue = true;
290  6                      return;
291  6                      end;
292  5                  call errptr(.('Length $'));
293  5                  acclen = 0;
294  5                  end;
295  4              end;
296  3          lastc = nextc;
297  3          nextc = gnc;

298  3          if token = ident then
299  3              do;
300  4              if nextc = '$' then nextc = 0; else
302  4              if not alphanum then
303  4                  return;
                    end; else

305  3          if token = number then
306  3              do;
307  4              if nextc = '$' then nextc = 0; else
309  4              if not hex then
310  4                  do; /* look for radix indicator */
311  5                  if (nextc='O') or (nextc='Q') then stype = octv; else
313  5                  if nextc = 'H' then stype = hexv;
                        if stype > 0 then nextc = 0; else
317  5                  if lastc = 'B' then
318  5                      do; acclen = acclen - 1; stype = binv;
321  6                      end; else
322  5                  if lastc = 'D' then
323  5                      do; acclen = acclen - 1; stype = decv;
326  6                      end; else stype = decv;
                        /* now convert the number and place into value */
328  5                      do;
329  6                      value = 0;
330  6                        do i = 1 to acclen;
331  7                        if (d := accum(i-1)) >= 'A' then
332  7                            d = d - 'A' + 10; else d = d - '0';
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
334  7                      if (b:=stype) <= d then token = errv;
336  7                      v = value; value = d;
338  7                        do while b <> 0;
339  8                        v = shl(v,1);
340  8                        if (b:=shr(b,1)) then
341  8                          do; value = value + v;
343  9                          if carry then token = errv;
345  9                          end;
346  8                        end;
347  7                      end;
348  6                    end;
349  5                  if token = errv then
350  5                      call errptr(,('Convert$'));
351  5                  token = number;
352  5                  return;
353  5                  end;
                      end; else

355  3              if token = string then
356  3                  do;
357  4                  if nextc = '''' then
358  4                      do; if (nextc:=gnc) <> '''' then return;
361  5                      end;
362  4                  if nextc = cr then
363  4                      do;
364  5                      call errptr(,('Quote  $'));
365  5                      return;
366  5                      end;
367  4                  end; else
                    /* must be special */
368  3                  return;
369  3              end; /* of do forever */
370  2          end gnt;

371  1      scan$ini: procedure public;
                  /* initialize scanner parameters */
372  2          value,nextc,token = 0;
373  2          continue = false;
374  2          end scan$ini;

375  1      end;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
8	0000H	2	A. . . . . WORD PARAMETER 9
5	0000H	2	A. . . . . WORD PARAMETER 6
22	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 23 24
26	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 27 29
75	00CDH	19	ABORT. . . . . PROCEDURE PUBLIC STACK=001AH 279
229	0296H	1	ACCLEN . . . . . BYTE PUBLIC 222 261 284 285 293 319 324 330
229	0297H	32	ACCUM. . . . . BYTE ARRAY(32) PUBLIC 284 285 331
258	06ADH	15	ALPHANUM . . . . . PROCEDURE BYTE STACK=0008H 302
230	0004H	1	B. . . . . BYTE PARAMETER AUTOMATIC 231 232 233
248	02B8H	1	B. . . . . BYTE 334 338 340
68	0000H	1	B. . . . . BYTE PARAMETER 69
246			BINV . . . . . LITERALLY 320
11	0000H		BOOT . . . . . PROCEDURE EXTERNAL(2) STACK=0000H 79 191
107	0000H	4	BUFF . . . . . BYTE BASED(BUFFA) ARRAY(4) 111 113 114
105	0004H	2	BUFFA. . . . . WORD PARAMETER AUTOMATIC 106 107 111 113 114
236	02B7H	1	C. . . . . BYTE 242 244
132	0004H	1	C. . . . . BYTE PARAMETER AUTOMATIC 133 136
83	028BH	1	C. . . . . BYTE 84 86
173	0291H	1	C. . . . . BYTE 179 180
108	028DH	1	C. . . . . BYTE 111 112 113
			CARRY. . . . . BUILTIN 343
66	0272H	8	CBUFF. . . . . BYTE ARRAY(8) PUBLIC 122 144 196 197 200
14	0004H	1	CHAR . . . . . BYTE PARAMETER AUTOMATIC 15 16
131	028FH	1	CHAR . . . . . BYTE 142 145 146 148 155 157 159 160 161 164
82	0143H	23	CHECKEOF . . . . . PROCEDURE BYTE STACK=0002H 91 103
36	0057H	19	CLOSE. . . . . PROCEDURE PUBLIC STACK=000AH
229	0295H	1	CONTINUE . . . . . BYTE PUBLIC 262 263 289 373
13			CR . . . . . LITERALLY 19 73 146 147 188 190 362
18	0013H	17	CRLF . . . . . PROCEDURE PUBLIC STACK=000EH 28 77
248	02BAH	1	D. . . . . BYTE 331 332 333 334 337
65	0000H	2	DBP. . . . . WORD 84 90 92 93 94 97 98 100 102 195
65	0021H	512	DBUFF. . . . . BYTE ARRAY(512) 84 90 93 94 97 98 195
31	0020H	1	DCNT . . . . . BYTE PUBLIC 34 38 54 186
246			DECV . . . . . LITERALLY 325 327
4			DEFBUF . . . . . LITERALLY 184
40	006AH	16	DELETE . . . . . PROCEDURE PUBLIC STACK=000AH
44	007AH	16	DISKREAD . . . . . PROCEDURE BYTE PUBLIC STACK=000AH 95
48	008AH	16	DISKWRITE. . . . . PROCEDURE BYTE PUBLIC STACK=000AH
60	0004H	2	DMA. . . . . WORD PARAMETER AUTOMATIC 61 62
67			EJECT. . . . . LITERALLY 73
68	0000H		EMITBYTE . . . . . PROCEDURE EXTERNAL(4) STACK=0000H
71	0000H		EMITCRLF . . . . . PROCEDURE EXTERNAL(5) STACK=0000H
67			EOFILE . . . . . LITERALLY 84 89 97 142 145 240
66	0271H	1	EOFSET . . . . . BYTE PUBLIC 85 88 193 239 278
74	027AH	17	ERRMSG . . . . . BYTE ARRAY(17) INITIAL 198 206 208 209 211 220
213	03BFH	65	ERRPTR . . . . . PROCEDURE PUBLIC STACK=001EH 292 350 364
66	0270H	1	ERRSET . . . . . BYTE PUBLIC 193 216 219
246			ERRV . . . . . LITERALLY 335 344 349
8	0000H	1	F. . . . . BYTE PARAMETER 9

COPYRIGHT © 1981
DIGITAL RESEAR
P. O. BOX 579
PACIFIC GROVE, CA
95
USER ==
233

```
   5  0000H    1  F. . . . . .   BYTE PARAMETER 6
  13            FALSE. . . . .  LITERALLY 128 168 175 193 263 373
  56  0004H    2  FCB. . . . . .  WORD PARAMETER AUTOMATIC 57 58
  52  0004H    2  FCB. . . . . .  WORD PARAMETER AUTOMATIC 53 54
  32  0004H    2  FCB. . . . . .  WORD PARAMETER AUTOMATIC 33 34
  44  0004H    2  FCB. . . . . .  WORD PARAMETER AUTOMATIC 45 46
  40  0004H    2  FCB. . . . . .  WORD PARAMETER AUTOMATIC 41 42
  36  0004H    2  FCB. . . . . .  WORD PARAMETER AUTOMATIC 37 38
  48  0004H    2  FCB. . . . . .  WORD PARAMETER AUTOMATIC 49 50
  13            FOREVER. . . . .  LITERALLY 281
 130  01FAH  164  GETLINE. . . . .  PROCEDURE STACK=001CH 202 238
 235  0413H   45  GNC. . . . . .  PROCEDURE BYTE STACK=0020H 267 297 359
  81  00E0H   99  GND. . . . . .  PROCEDURE BYTE STACK=00EH 142 164
 247  0440H  560  GNT. . . . . .  PROCEDURE PUBLIC STACK=0024H
 252  0681H   27  HEX. . . . . .  PROCEDURE BYTE STACK=0006H 309
 246            HEXV . . . . . .  LITERALLY 314
 248  02B9H    1  I. . . . . .  BYTE 330 331
 215  0292H    1  I. . . . . .  BYTE 222 223
 173  0290H    1  I. . . . . .  BYTE 174 175 178 179 196 197 198
 119  028EH    1  I. . . . . .  BYTE 124 125
 108  028CH    1  I. . . . . .  BYTE 109 110 111 113 114 115
 228            IDENT. . . . . .  LITERALLY 269 298
  64  0000H   33  IFCB . . . . . .  BYTE ARRAY(33) EXTERNAL(3) 64 95 182 183 185
  64  0020H    1  IFCBR. . . . . .  BYTE EXTERNAL(3) AT 194
 105  015FH   89  INCLINE. . . . .  PROCEDURE STACK=0004H 144 220
                  LAST . . . . . .  BUILTIN 90 174 196 200 285
 248  02BCH    1  LASTC. . . . . .  BYTE 296 317 322
  65  026DH    1  LBLEN. . . . . .  BYTE PUBLIC 123 124 141 166 237
  65  026CH    1  LBP. . . . . .  BYTE 134 136 140 151 166 170 222 237 242 243
  65  0221H   75  LBUFF. . . . . .  BYTE ARRAY(75) PUBLIC 125 134 136 242
                  LENGTH . . . . . .  BUILTIN 93 98 134 195
 255  069CH   17  LETTER . . . . . .  PROCEDURE BYTE STACK=0002H 259 268
  13            LF . . . . . .  LITERALLY 20 73 145 188 190
  65  026EH    1  LINEOUT. . . . . .  BYTE PUBLIC 168
  65  026FH    1  LINEREADY. . . . .  BYTE PUBLIC 120 128 167 193
  67            LIT. . . . . .  LITERALLY 67 228 246
   2            MAJOR. . . . . .  LITERALLY 73
  52  009AH   19  MAKE . . . . . .  PROCEDURE PUBLIC STACK=000AH
   2            MINOR. . . . . .  LITERALLY 73
   5  0000H       MON1 . . . . . .  PROCEDURE EXTERNAL(0) STACK=0000H 16 24 42 58 62
   8  0000H       MON2 . . . . . .  PROCEDURE BYTE EXTERNAL(1) STACK=0000H 34 38 46 50 54
                  MOVE . . . . . .  BUILTIN 183
  75  0004H    2  MSG. . . . . .  WORD PARAMETER AUTOMATIC 76 78
 213  0004H    2  MSG. . . . . .  WORD PARAMETER AUTOMATIC 214 221
 229  0293H    1  NEXTC. . . . . .  BYTE PUBLIC 250 253 256 266 267 272 275 282 284 286
                  297 300 301 307 308 311 313 316 357 359 362 372
 228            NUMBER . . . . . .  LITERALLY 271 305 351
 249  0670H   17  NUMERIC. . . . . .  PROCEDURE BYTE STACK=0002H 253 259 270
 246            OCTV . . . . . .  LITERALLY 312
  32  0044H   19  OPEN . . . . . .  PROCEDURE PUBLIC STACK=000AH 185
   3  0006H   26  PARM . . . . . .  BYTE ARRAY(26) PUBLIC 174 175 180
  64  006DH    8  PARMS. . . . . .  BYTE ARRAY(8) AT ABSOLUTE 177 179
  26  0034H   16  PRINT. . . . . .  PROCEDURE PUBLIC STACK=0014H 78 122 188 201 211 221
  14  0000H   19  PRINTCHAR. . . . .  PROCEDURE PUBLIC STACK=000AH 19 20 125 189 224 226
  22  0024H   16  PRINTN . . . . . .  PROCEDURE PUBLIC STACK=000AH 29 190
 132  029EH   30  PUTCHAR. . . . . .  PROCEDURE STACK=0004H 147 150 152 161
 230  0400H   19  PUTEMITCHAR. . . .  PROCEDURE STACK=0008H
```

COPYRIGHT © 1981
DIGITAL RESEAR
P.O. BOX 5
PACIFIC GROVE, C
SER. #

```
 118  0138H   66  PUTLINE. . . . . PROCEDURE PUBLIC STACK=0018H       139  205  218
  56  00ADH   16  RENAME . . . . . PROCEDURE PUBLIC STACK=000AH
 371  06BCH   20  SCANINI. . . . . PROCEDURE PUBLIC STACK=0002H
   1  0000H       SCANMOD. . . . . PROCEDURE STACK=0000H
  60  00BDH   16  SETDMA . . . . . PROCEDURE PUBLIC STACK=000AH        94  184
 172  02BCH  227  SETUP. . . . . . PROCEDURE PUBLIC STACK=0020H
                  SHL. . . . . . . BUILTIN        339
                  SHR. . . . . . . BUILTIN        340
 228              SPECIAL. . . . . LITERALLY      277
 228              STRING . . . . . LITERALLY      274  287  355
 248  02F9H    1  STYPE. . . . . . BYTE      261  312  314  315  320  325  327  334
  67              TAB. . . . . . . LITERALLY      148
  73  0000H   41  TITLE. . . . . . BYTE ARRAY(41) DATA      201
 229  0294H    1  TOKEN. . . . . . BYTE PUBLIC        264  265  269  271  274  277  287  298  305  3
                                     344  349  351  355  372
  13              TRUE . . . . . . LITERALLY       85  167  180  219  281  289
 248  0004H    2  V. . . . . . . . WORD      336  339  342
 229  0062H    2  VALUE. . . . . . WORD PUBLIC        329  336  337  342  372
 204  039FH   32  WRITERRS . . . . . PROCEDURE PUBLIC STACK=001CH
```

MODULE INFORMATION:

```
  CODE AREA SIZE     = 06D0H   1744D
  CONSTANT AREA SIZE = 0180H    384D
  VARIABLE AREA SIZE = 02BDH    701D
  MAXIMUM STACK SIZE = 0024H     36D
  491 LINES READ
  0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE TOKEN
OBJECT MODULE PLACED IN TOKEN.OBJ
COMPILER INVOKED BY:  :F0:TOKEN.PLM XREF OPTIMIZE(2) DATE(APRIL 6, 1982)


  1          token:
             do;
             $include (:f1:lit.plb)
         =   /* literal declarations for basic tokens */
  2   1  =   declare
         =       lit literally 'literally',
         =       tiden  lit '01', tnumb  lit '02', tstrng lit '03', tspecl lit '04',
         =
         =       /* token# is the base token number for each set of tokens */
         =       token2 lit '140', token3 lit '160', token4 lit '180',
         =       token5 lit '200', token6 lit '210', token7 lit '220',
         =       token8 lit '230',
         =       tokenm lit '7',
         =
         =       /* single character tokens ! " # $ % & ( ) * + , - . / : ; < = > ?
         =       are all represented by their ascii values */
         =
         =       /* special two character sequences (started by single character) */
         =
         =       /* two character tokens: */
         =
         =       /* three character tokens: */
         =
         =       /* four character tokens: */
         =
         =       /* five character tokens: */
         =       tdisks lit '200', tendef lit '201',
         =
         =       /* six character tokens: */
         =       tmaclib lit '210',
         =
         =       /* seven character tokens: */
         =       tdiskdef lit '220';

  3   1      declare
                 cr lit '0dh';

             /* token tables given below are each addressed by tokadr(length), and
             each start with the number of items in the table of that particular
             length.  the tables are in sorted order so that they could be searched
             with a binary search algorithm (sequential is used at this time) */

  4   1      declare
                 /* tokens are not listed for length 0,1 */
                 tok0 (*) byte data(0), tok1 (*) byte data(0),
                 /* single character tokens are given by their ascii representation */
                 tok2 (*) byte data
                 (0),
                 tok3 (*) byte data
                 (0),
                 tok4 (*) byte data
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                (0),
                tok5 (*) byte data
                (2,'DISKS','ENDEF'),
                tok6 (*) byte data
                (1,'MACLIB'),
                tok7 (*) byte data
                (1,'DISKDEF');

 5  1      declare
            /* index to base address of each table */
            tokadr (*) address data
            (.tok0,.tok1,.tok2,.tok3,.tok4,.tok5,.tok6,.tok7),

            /* starting token number for each length */
            tokbas (*) byte data
            (0,0,token2,token3,token4,token5,token6,token7);

        /* external declarations */
 6  1      declare
            nextc      byte external,   /* next char (lookahead) */
            token      byte external,   /* current token */
            continue   byte external,   /* true for long idents, strings */
            acclen     byte external,   /* accumulator length */
            accum(32)  byte external;   /* actual characters scanned */

 7  1      gnt: procedure external;
            /* produces token = tiden, tnumb, tstrng, or tspecl */
 8  2          end gnt;

 9  1      scan: procedure public;
            /* scan produces the actual token number for each item returned by
            gnt (get next token).  in the case of identifiers and special chars,
            the token tables are searched before returning the token number */

10  2          nextchar: procedure(lookchr) byte;
11  3              declare lookchr byte;
                /* nextchar is used to look ahead for special two character
                sequences.  if 'lookchr' is found, then nextc is zeroed and true is
                returned from the call */
12  3              declare tf byte;
13  3              if (tf := nextc = lookchr) then nextc = 0;
15  3              return tf;
16  3              end nextchar;

17  2          declare /* local variables for the token matching */
                ta address,       /* set to beginning of string of symbols */
                tokstr based ta (tokenm) byte, /* string template at ta */
                n byte,           /* number of symbols remaining to scan in string */
                i byte;           /* index used while matching characters */

18  2          call gnt; /* sets external variables */
19  2          if token = tstrng or token = tnumb then return;

            /* otherwise token = tspecl or token = tiden */

21  2          if token = tspecl then
22  2              do;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                    /* may be a comment */
 23   3             if accum(0) = ';' then
 24   3                 do while accum(0) <> cr;
 25   4                 call gnt;
 26   4                 end;
 27   3             token = accum(0);
 28   3             return;
 29   3             end;

 30   2         if acclen > tokenm then
                     /* cannot be a reserved word */
 31   2              return;

 32   2         ta = tokadr(acclen);  /* ta is set to the base string to match */
 33   2         n  = tokstr(0);   /* n is the number of symbols in the string */

                  /* token must be set to tiden at this point */
 34   2         token  = tokbas(acclen);  /* base token number */

 35   2             do while n > 0; /* more match attempts */
 36   3             n = n - 1;  i = acclen;
 38   3                 do while i > 0 and accum(i-1) = tokstr(i);
                         /* one more character has been matched */
 39   4                 i = i - 1;
 40   4                 end;
 41   3             if i = 0 then /* a complete match was found */
 42   3                 return;
                     /* current token does not match, try again */
 43   3             token = token + 1; /* move to next token in sequence */
 44   3             ta = ta + acclen; /* base address advanced to next item */
 45   3             end;
                   /* cannot find the token, leave 'token' set to tiden */
 46   2         token = tiden;
 47   2         return;
 48   2         end scan;
 49   1     end;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
6	0000H	1	ACCLEN . . . . . BYTE EXTERNAL(3) 30 32 34 37 44
6	0000H	32	ACCUM. . . . . BYTE ARRAY(32) EXTERNAL(4) 23 24 27 38
6	0000H	1	CONTINUE . . . . . BYTE EXTERNAL(2)
3			CR . . . . . LITERALLY 24
7	0000H		GNT. . . . . PROCEDURE EXTERNAL(5) STACK=0000H 18 25
17	0004H	1	I. . . . . BYTE 37 38 39 41
2			LIT. . . . . LITERALLY 2 3
10	0004H	1	LOOKCHR. . . . . BYTE PARAMETER AUTOMATIC 11 13
17	0003H	1	N. . . . . BYTE 33 35 36
6	0000H	1	NEXTC. . . . . BYTE EXTERNAL(0) 13 14
10	00CDH	33	NEXTCHAR . . . . . PROCEDURE BYTE STACK=0004H
9	0000H	205	SCAN . . . . . PROCEDURE PUBLIC STACK=0004H
17	0000H	2	TA . . . . . WORD 17 32 33 38 44
2			TDISKDEF . . . . . LITERALLY
2			TDISKS . . . . . LITERALLY
2			TENDEF . . . . . LITERALLY
12	0002H	1	TF . . . . . BYTE 13 15
2			TIDEN. . . . . LITERALLY 46
2			TMACLIB. . . . . LITERALLY
2			TNUMB. . . . . LITERALLY 19
4	0010H	1	TOK0 . . . . . BYTE ARRAY(1) DATA 5
4	0011H	1	TOK1 . . . . . BYTE ARRAY(1) DATA 5
4	0012H	1	TOK2 . . . . . BYTE ARRAY(1) DATA 5
4	0013H	1	TOK3 . . . . . BYTE ARRAY(1) DATA 5
4	0014H	1	TOK4 . . . . . BYTE ARRAY(1) DATA 5
4	0015H	11	TOK5 . . . . . BYTE ARRAY(11) DATA 5
4	0020H	7	TOK6 . . . . . BYTE ARRAY(7) DATA 5
4	0027H	8	TOK7 . . . . . BYTE ARRAY(8) DATA 5
5	0000H	16	TOKADR . . . . . WORD ARRAY(8) DATA 32
5	002FH	8	TOKBAS . . . . . BYTE ARRAY(8) DATA 34
1	0000H		TOKEN. . . . . PROCEDURE STACK=0000H
6	0000H	1	TOKEN. . . . . BYTE EXTERNAL(1) 19 21 27 34 43 46
2			TOKEN2 . . . . . LITERALLY 5
2			TOKEN3 . . . . . LITERALLY 5
2			TOKEN4 . . . . . LITERALLY 5
2			TOKEN5 . . . . . LITERALLY 5
2			TOKEN6 . . . . . LITERALLY 5
2			TOKEN7 . . . . . LITERALLY 5
2			TOKEN8 . . . . . LITERALLY
2			TOKENM . . . . . LITERALLY 17 30
17	0000H	7	TOKSTR . . . . . BYTE BASED(TA) ARRAY(7) 33 38
2			TSPECL . . . . . LITERALLY 21
2			TSTRNG . . . . . LITERALLY 19

CP/M-86 v.1.1 (Vol. II)
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # C81-162

MODULE INFORMATION:

CODE AREA SIZE = 00EEH 238D

```
CONSTANT AREA SIZE = 0037H     55D
VARIABLE AREA SIZE = 0005H      5D
MAXIMUM STACK SIZE = 0004H      4D
144 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____